

Entropy Non-increasing Games for the Improvement of Dataflow Programming

Norbert Bátfai, Renátó Besenczi, Gergő Bogacsovics,
Fanny Monori*

February 16, 2017

Abstract

In this article, we introduce a new conception of a family of esport games called Samu Entropy to try to improve dataflow program graphs like the ones that are based on Google’s TensorFlow. Currently, the Samu Entropy project specifies only requirements for new esport games to be developed with particular attention to the investigation of the relationship between esport and artificial intelligence. It is quite obvious that there is a very close and natural relationship between esport games and artificial intelligence. Furthermore, the project Samu Entropy focuses not only on using artificial intelligence, but on creating AI in a new way. We present a reference game called Face Battle that implements the Samu Entropy requirements.

Keywords: *esport, TensorFlow, computation graphs, gamification, robopsychology.*

1 Introduction

By now, playing massively multiplayer online (MMO) games has become a part of daily routine. Many computer game competitions and a whole new industry called esport is being built on the MMO game concept. The development of esports has been living in close symbiosis with the evolution of artificial intelligence research. The collaboration between Google DeepMind and Blizzard Entertainment¹ can also be interpreted as a next milestone in the development of artificial intelligence. In the present paper, we are going to introduce a new esport game family concept called Samu Entropy (or ESAMU, for short) [Bát17b], [BBL⁺17]. The ESAMU concept is outlined in the English translation of its developer’s guide² [BBS16].

Automated development of artificial convolutional neural network architectures is a hot topic in machine learning (ML). For example, papers [ZL16] and

*All authors are with the Department of Information Technology, University of Debrecen, Hungary, {batfai.norbert, besenczi.renato}@inf.unideb.hu

¹DeepMind and Blizzard to release StarCraft II as an AI research environment, <https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/>

²Pre-release of the Samu Entropy documentation, <https://github.com/nbatfai/SamuEntropy/releases/tag/v0.0.1>.

[BGNR16] present reinforcement learning approaches to develop the computing models, where the former one is based on TensorFlow (TF) computational graphs [AAB⁺16] and the latter one uses Q-learning. The authors of paper [VSR⁺16] reported an evolutionary computing approach. As the main result of this paper, we introduce a new esport based approach to refine TensorFlow computational graphs. Specifically, we introduce a concept of an esport game with which a player will be able to search better neural network architectures.

To embed a scientific challenge into a computer game is a well-known idea. The examples for this have ranged from science fiction stories to hard-core science (see for example the science fiction book SGU [Swa09, pp. 32] or the scientific puzzle game called *Foldit* [CTB⁺10]). It is important to note that ESAMU is not a typical human-computing [TSBW16] or citizen science [PHS15] game but, in contrast with the previously mentioned games, it is intended to become a killer application. At the present time, applications based on machine learning are becoming closer and closer to the normal everyday life. Nonetheless, developing deep learning applications is still a scientific task which requires highly trained and experienced professionals from this field. In our vision, an application, which implements the ESAMU concepts, can take the whole machine learning science closer to the public by giving a simplified interface for AI algorithms and methods. Certainly, killer apps cannot be “developed”, as nobody can decide whether a computer program becomes successful or not. But, with developing such an application we hope that it will gain a high level of attention, not only from the scientific community, but from a large segment of the public. Accordingly, ESAMU is not a specific game but only a specification set for the games to be developed.

It seems that TensorFlow may be the Pax Romana of machine learning programming (paper [Gol16] shows a timeline of machine learning programming languages and the second one of the milestone works [MKS⁺15] and [SHM⁺16] used TensorFlow³). TensorFlow is an open source heterogeneous machine learning software platform for running TensorFlow computational graphs on CPU, GPU, TPU, Android or iOS. By using computational graphs in a central role, TensorFlow (Python or C++ based) API programming can be regarded as a kind of classical dataflow programming [Sou12], [Kah74].

In the following, a short description of the ESAMU concept [BBS16] will be presented by way of a specific game called Face Battle. We focus on the question of embedding a TensorFlow computational graph into a computer game. At the current stage of the research, the graph of MNIST⁴ tutorial example [Aut17] will be used.

2 Face Battle

In the ESAMU esport family, every game will be developed following five archetypes, so all ESAMU games must implement these behaviors. Every game will implement some sort of machine learning task, so we require a dedicated archetype for implementing AI methods. This first archetype is called “Samu, the Brain”. Practically, this archetype uses currently available machine learning

³See <https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html>

⁴MNIST For ML Beginners, <https://www.tensorflow.org/tutorials/mnist/beginners/>

algorithms and paradigms (with the primary focus on deep learning), and it can be extended in the future. Because the proper usage and sometimes even the understanding of these machine learning primitives can be difficult for non-experts, we use the “Gréta, the Builder” (GTB) archetype for fine-tuning end-to-end machine learning “flows”. In our vision, one implementation of this archetype could be an application where the user can fine tune the dataflow graph of Inception v3 [SVI⁺15] if the player wants to create an application suitable for object recognition. The apps implementing Samu and Gréta are universal. That means that these two archetypes are independent from a specific implementation of ESAMU, and can be used in other ESAMU games. The apps from the archetype of Samu execute TF graphs and the apps from the archetype of Gréta give the possibility to edit and fine-tune that TF graph. The “Nándi, the Teacher” archetype is applied for the implementation of software that can use supervised learning. The archetype called “Matyi, the Hunter” consists of software that provide perception and intervention, e.g. in the Face Battle application, collecting images of faces can be considered as a software for this archetype. The archetype “Erika, the Fighter” realizes the competition aspect of ESAMU games, e.g. in the Face Battle game, comparing the accuracy of face recognition can be considered as a simple implementation of this archetype, but its main motivation is to provide an esports experience for ESAMU games (i.e. esports competitions that can be organized in an arena).

In the rest of this section, we give a description about how we use these archetypes in the Face Battle game.

2.1 Samu, the Brain

The main purpose of this archetype is to execute TF graphs. Our main question is, what is the most suitable ML architecture for a corresponding task?

There are several well-known ML architectures that perform well on various datasets. In the past few years, convolutional neural networks (CNN) [LBBH98] have made a reasonable break-through in the field. One of the earliest works that had a great impact was reported by Krizhevsky et al. in [KSH12]. This architecture used five convolutional layers and three fully connected layers and introduced several new solutions (e.g. “dropout”). Another approach is the Inception architecture that is first introduced in [SLJ⁺14]. This architecture was 22 layers deep, which ignited the spread of very deep CNNs. Later, the Inception architecture was improved to achieve better performance [SVI⁺15]. The aforementioned solutions were trained and tested on the ImageNet open dataset. Face recognition is also considered as a classical AI task. One example is DeepFace [TYRW14] and its improved version FaceNet [SKP15], both were trained and tested on Labeled Faces in the Wild and YouTube Faces dataset, and set the state-of-the-art performance in its time of publishing. Since the dataflow graph of these architectures are huge (tens of thousands of nodes in some cases), we are planning to provide a basic version that the player can fine-tune, so they do not need to build it from scratch. In the game Face Battle, this could be the dataflow graph of FaceNet.

These examples show us that some sort of a natural evolution can be seen in machine intelligence. This evolution is obviously powered by science and the natural need to create more and more efficient AI algorithms. But, this creation requires highly trained and experienced professionals. With the ESAMU

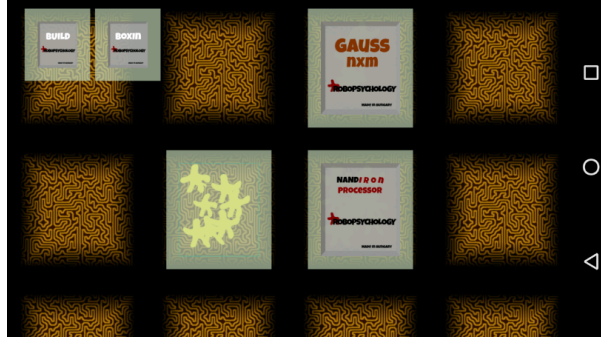


Figure 1: A screenshot of the app Motherboard Builder, which is a prototype for the GTB archetype.

concept, we try to create an architecture that can facilitate this evolution by connecting non-expert gamers to the AI research communities.

2.2 Gréta, the Builder

In the game Face Battle, the Motherboard Builder is our first rapid prototype⁵ for the GTB archetype. One key point of this motherboard approach is that we can place various objects related to computing on its surface. The idea came from the appearance and functionality of a motherboard of a desktop computer. Just like a real one, it can be equipped with several different “devices”, which can be changed, modified or upgraded later. In the aspect of an ESAMU game, these objects can be TF nodes (e.g. a CNN layer). It is not rare for a TensorFlow graph to have several thousand nodes. For example, the model of the TF tutorial example MNIST contains 137 nodes, but more sophisticated ones can contain tens of thousands of nodes. Building this from scratch on a screen or reviewing such a huge graph is impossible. Our aim is to simplify the visualization of complex ML architectures, so it would be easier to view the whole graph and fine-tune individual nodes. One of our main research question is: how can a complex TensorFlow graph be visualized so that the possibility to fine-tune individual nodes could remain. In section 3.1, we will present some more ideas about the GTB archetype. Figure 1 shows a screenshot of Motherboard Builder.

2.3 Nándi, the Teacher

This archetype provides supervised learning in games where it is necessary. In the game Face Battle, this archetype will implement a set of possible annotations and corrections related to the images taken by players. This will allow players to connect persons, emotions, locations, etc. to images, and to correct any possible misclassification (e.g. emotion recognition).

⁵See in repository https://github.com/nbatfai/SamuEntropy/tree/master/FACE_BATTLE/Greta/GretaTheBuilder

2.4 Matyi, the Hunter

This archetype is responsible for perception, data collection and, in some cases, intervention. In Face Battle, it will provide photo acquisition. It is important to implement this archetype in Face Battle, because we want to collect images about faces only. Therefore, some sort of pre-processing is required, e.g. face segmentation or alignment. In addition, this part of the software will have a function to recognize emotions too.

2.4.1 Face Battle Dataset

For every ESAMU game, distinct datasets are required to perform machine learning tasks. Using open datasets for testing face recognition methods are becoming a common practice, the number of open datasets is growing (see e.g.: [LMHR⁺16], [YLLL14], [LLWT15], [ZZWS12], [KSSMB16], [NW14]). In our basic concept, every game has its own dataset and every dataset is open, moreover, every subset is linked to a certain player, who collected the corresponding data, via a social network profile. To the best of our knowledge, this will be the first public dataset where images are linked to people, except for datasets related to celebrities.

2.5 Erika, the Fighter

This archetype will offer the possibility of building a competition around Face Battle. This will be the part of the game where the players can compare each other's "learning architecture", or more precisely, the efficiency (i.e. the error rate) of their face classification procedure.

One use could be the following: players will give each other some pictures about themselves. After learning these pictures, the accuracy of classification of other pictures of the same player will be compared. The player with the better accuracy wins the battle.

3 Entropy Non-increasing Interfaces of Games

First of all, it should be noted that the entropy non-increasing property of elements of the game interface should be understood only as an intuitive metaphor, because the entropy of game's display is nearly independent from the order of what is viewed in the display (see [Pen16, pp. 402] for an analogous example of a similar situation). The inspiration behind the idea of the game having an entropy non-increasing property is RTS (Real-Time Strategy) games we play. Exemplary well-known RTS games are *Age of Empires*⁶, *Warcraft*⁷, *StarCraft*⁸, or the *Cossacks*⁹ series, which were among the first generation of RTS games. Newer games like *Clash of Clans*¹⁰, or the open-source *0 A.D.*¹¹ also have great player base. The environment in these games may differ somewhat, but the

⁶<https://www.ageofempires.com/>

⁷<https://worldofwarcraft.com>

⁸<http://www.battle.net/sc2>

⁹<http://gsc-game.com/>

¹⁰<http://supercell.com/en/games/clashofclans/>

¹¹<https://play0ad.com/>, <https://github.com/0ad/0ad>



Figure 2: An illustration of “entropy” non-increasing property with two moments in the game *0 A.D.*

main tasks a player can perform are roughly the same. In an RTS game, the players usually start with an empty map without buildings and full of resources. Then by collecting these resources, the player can build more buildings, create playable units, improve their structures and generally move forward in the game. While playing with RTS games one can observe that the more the game progresses the more arranged it is. We believe that the game has its “entropy” at the highest in the beginning of the game with the empty map. With every “good” move the player makes the more ordered it becomes, so in our interpretation its “entropy” is decreasing. By good move, we intuitively mean a constructive event made by the player that moves him closer to winning. In figure 2 we illustrate this idea with two moments from the *0 A.D.* game. We think in some cases this non-increasing property can also be perceived visually, as the map of the game looks more arranged as the game progresses. The part of the game that implements the GTB archetype will have functions that are somewhat similar to building and creating activities in RTS games.

3.1 Towards the Gamification of GTB

The main purpose of the GTB archetype is to answer the research question: how can a TensorFlow computational graph be converted into the interface of a massively multiplayer game. At the current stage of the research, it seems that it is reasonable to focus on some MMORTS (Massively Multiplayer Online

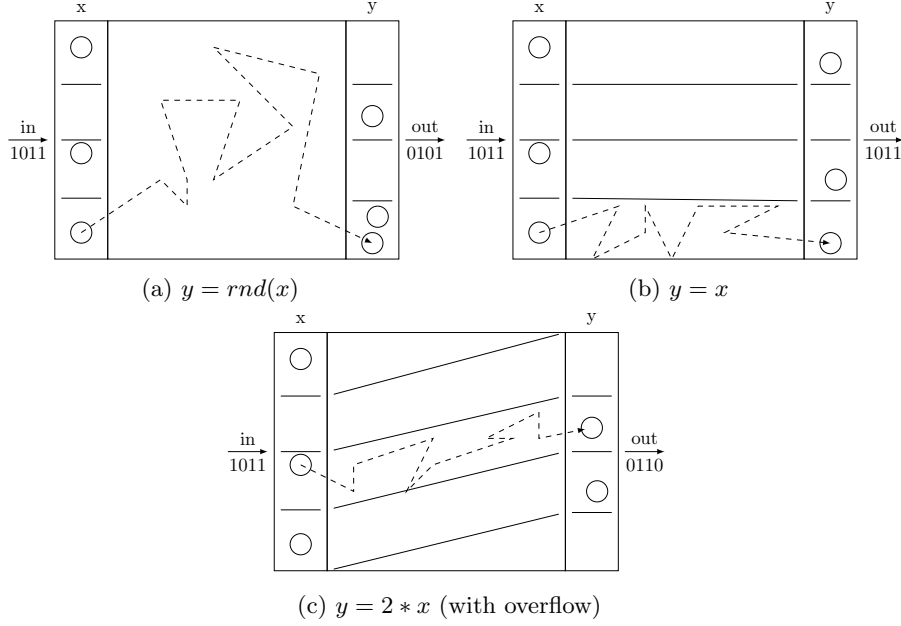


Figure 3: This figure shows a fictional computing device where the programming is done by arrangement of the box. The first box (an idealization of a computer program) has no internal walls. The second one contains 3 internal walls and the third one contains 4 internal walls. Accordingly, it follows that the other two programs have smaller entropy than the first one.

Real-Time Strategy) game to be developed from scratch.

In a typical RTS game, the elements of the game (e.g. buildings, army, town hall and other items) will become more and more sophisticated (or more complex) as times goes on. The ESAMU specification suggests measuring the development of these items in bits. From our point of view, the user interface of the game to be developed is a different visualization of a corresponding Tensor-Flow computation graph. After all, we would like to measure the goodness (the level of development) of elements of a computation graph. So, we would like to measure the goodness of the source code of pieces of the graph. First, we focus on the development of the whole source code as execution time goes on¹².

Let us start with the question, how can we measure the order of a source code? For example, how can we measure the order of a C or Python source code? There are several classical complexity measures, for example the cyclomatic complexity [McC76], but these can measure only some features of the source code. The cyclomatic complexity measures how readable the investigated source code snippet is but it cannot tell us about the real goodness of the code. That is why these measures cannot be considered objective. For a totally objective measure we could use some Kolmogorov complexity based measure, but it would not be computable due to the fact that the Kolmogorov complexity is not recursive (here it should be noticed that the similarity metric [LCL⁺04] is

¹²In our present approach, the changing neural network weights are considered as part of the source code.

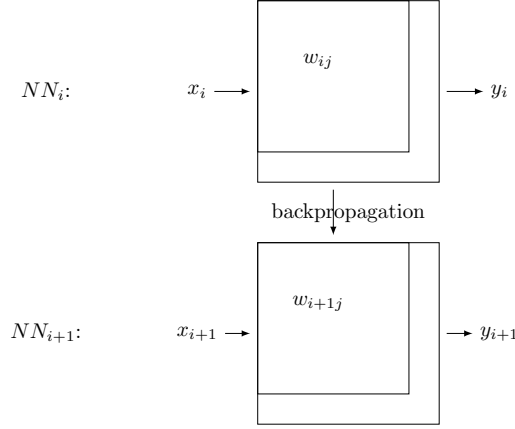


Figure 4: This figure shows a neural network between two consecutive training steps.

a Kolmogorov complexity based measure and it is computable for example with the use of CompLearn [CV05]). Another possibility is to investigate sources as a directed graph of generalized function calls¹³. In this case, the generalized functions can be sorted by their PageRank [PBMW99] values. But this order can make a recursive statement: a function that is called by better functions is better¹⁴.

How can we measure the development of computer programs? As it is well known, the Kolmogorov complexity has a strong relationship with the entropy [Lov94, pp. 69]. Consider the fictional computing device shown in figure 3. In these boxes, the programming is implemented by the internal arrangements of walls. The input particles are prepared on the left side and the output will appear on the right side as the time goes on¹⁵. Using the notation of figure 3, $K(x \mapsto \text{rnd}(x)) > K(x \mapsto x) \simeq K(x \mapsto 2x)$ intuitively shows that the entropy changes in the same direction as the Kolmogorov complexity. But these and similar imaginable measures tell us nothing about the questions: does the code meet its requirements or does the code fit for a particular purpose. This is why the test-driven development methodology uses more or less independent tests in today's software engineering practices.

In contrast, in the neural computing paradigm the evaluation of the fitness of the programs for a particular purpose (aka the artificial neural networks) is an essential feature. Consider the training and testing of the networks, they are investigated roughly in the same fashion. For example, in a classification task, if we use TensorFlow, we will typically examine accuracy curves in TensorBoard. Consider a practical question: after 10 training steps, is the complexity of a neural network the same as after 1000 steps? Based on figure 4, it is clearly observed that $K(NN_{i+1}) \leq |x_i| + |NN_i|$. If we suppose that the initial weight vector w_{0j} is selected from a random distribution, then the equation suggests

¹³For example, as like in C++ $a+b$; \rightarrow operator+(a, b); and in a similar way it can be imagine that $\text{for}(\text{stat1}; \text{expr1}; \text{expr2}) \text{ stat2}; \rightarrow \text{statementfor}(\text{stat1}, \text{expr1}, \text{expr2}, \text{stat2});$ and so on.

¹⁴We had made similar measurements with running codes using AspectJ in paper [Bát11].

¹⁵Because of the probability nature of this computing device, we cannot give an accurate upper bound of its execution time.

that the complexity (together with the entropy) is the biggest at the beginning. Both this and the previous figure contradict with our naive intuition that programming increases complexity. Therefore, in GTB, we must try to measure the sophistication level of the program GTB based on its fitness for a particular purpose. We will try to convert the usually applied accuracy into a measure that could be computed in bits. Our vision is to create the roots of a new type of gamification by using some similar measure that allows casual gamers to participate in the revolution of artificial intelligence. The idea is to assign the measured values to the elements of the interface in GTB. The first suggested theme for the game interface is the Motherboard Builder introduced in section 2.2. An imaginary element that has already been closely integrated with this theme can be seen in figure 5. This figure shows an abstraction of a processor that has 2.6 bits of information accuracy.

Information Accuracy

At this point, we present an example to illustrate how the usual accuracy can be converted into information. Let us consider the following example¹⁶.

Let $Y_- = (y_{-i})$ denote the sequence of labels of input images where $y_{-i} \in \{0, 1\}^n$ is a one-hot vector corresponded to an input image¹⁷. And let $Y = (p_i)$ denote the probabilities of classifications where $p_i \in \mathbb{R}^n$ such that $\sum_{j=1}^n p_i(j) = 1$. Then, using the entropies of labels $E = (e_i)$, $e_i \in \mathbb{R}$, $e_i = -\sum_{j=1}^n p_i(j) \log_2 p_i(j)$ we can define the information accuracy as follow $infoacc = \sum_{i=1}^m a_i e_i$ where

$$a_i = \begin{cases} -1 & \text{if } y_i \neq y_{-i} \\ 1 & \text{otherwise} \end{cases}$$

that is a_i simply shows that the classification of the i -th image is good or bad and m is the number of classified images in the sequence Y_- . Or to be more readable and precise

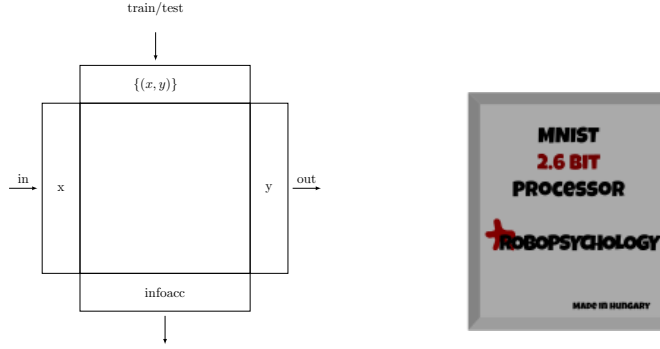
```
plogp = tf.multiply(prob, tf.log(prob)/tf.log(2.0))
plogpsum = -tf.reduce_sum(plogp, 1, keep_dims=True)
plusminus = tf.cast(correct_prediction,
    tf.float32)*2.0-1.0
plusminus = tf.reshape(plusminus, [batchsize, 1])
info = tf.multiply(plogpsum, plusminus)
infoacc = tf.reduce_sum(info) /
    tf.cast(batchsize, tf.float32)
```

where the tensor *prob* holds the vectors of probabilities of classifications and *batchsize* is the number of images.

Figure 6 shows some usual accuracy curves with *batchsize* of 10, 100, 1000 and 10000. The corresponding information accuracy curves are shown in figure 7. We can observe that the shapes of the curves are the same, but the scale has changed. So, the “learning” of the neural network architecture is followed by the information accuracy of the learning procedure.

¹⁶It can be seen in detail in the forked TensorFlow repository <https://github.com/nbatfai/tensorflow/blob/master/tensorflow/examples/tutorials/mnist>. The presented Python code snippet can be found in the file called `mnist_softmax_esmu.py`.

¹⁷In this example n equals 10 due to a label is a digit between 0 and 9.



(a) This figure shows the typical use cases of a neural network based program where we replace the usual accuracy by information accuracy.

(b) This is an imaginary chip which represents the gamification of the MNIST graph of the TensorFlow tutorial.

Figure 5: A gamification of a whole network is shown in general in 5a and a specific gamification element is shown in 5b. The label ROBOPSYCHOLOGY on the chip is only a design element that suggests that there is a higher interpretation level of our work. See [Bát17a] and [BB17] for more detail.

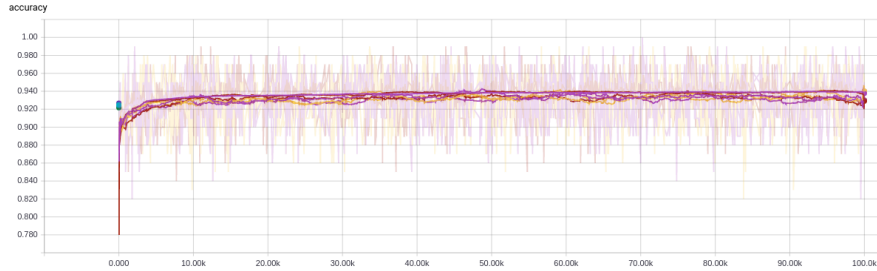


Figure 6: This figure shows the usual accuracy curves of MNIST tutorial in TensorBoard.

4 Conclusion

From a pure theoretical point of view, our present work can be regarded as a set of preliminary studies regarding the following questions: What notion of complexity levels are we looking for to the gamification of computing graphs? How can we measure the aliveness of the code? How could the code resist against increasing entropy? The latter two questions have already focused on the approach introduced by Erwin Schrödinger in his famous book [Sch44] to investigate living systems. To apply his ideas in the field of computer science seems a very interesting challenge¹⁸. The thought experiments shown in figure 3 and 4 contradicts our intuitive expectation that programming increases complexity (entropy), therefore in this sense, programming is similar to entropy

¹⁸In the terminology of Turing machines, we have already formulated a similar question in <https://github.com/nbatfai/AlgorithmicFractals/tree/master/manuscript>.

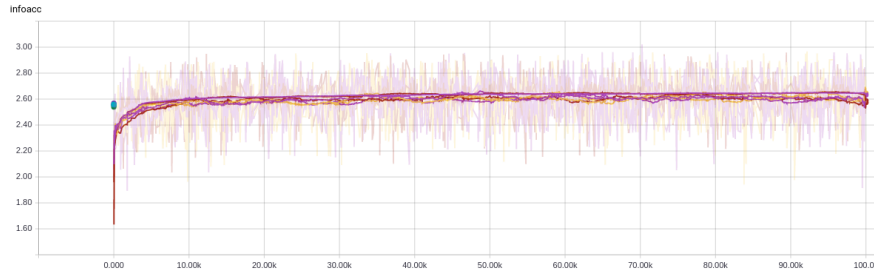


Figure 7: This figure shows the information accuracy curves corresponding to the curves in figure 6.

non-increasing behavior of living systems.

The work described in this paper has formed and crystallized our main research question: how can a TensorFlow computational graph be visualized on the surface of a massively multiplayer game? Simplifying a TF graph without losing important components and providing a possibility to fine-tune it by non-expert gamers at the same time seems an interesting challenge. We think, answering this question is crucial to making a big leap towards the manifestation of the conception of widely open AI. Google with the TensorFlow API have simplified the prototyping of ML architectures and have made it easier to experiment with new ideas, even though it requires broad knowledge in AI methods and algorithms. In our vision, a powerful tool that can speed up the evolution of ML by widening the group of people who working with it, can be applications that implement the ESAMU concept. By playing these games, which we consider as an embryonic form of robopsychology, we may get closer to the creation of The Artificial Intelligence in its universal form.

Acknowledgment

The authors would like to thank the students of the BSc course of “High Level Programming Languages” in the fall semester of 2016/2017 at the University of Debrecen and the community members of the group UDPROG¹⁹ for their interest and for their testing and contribution to the repository of ESAMU [Bát17b]. Special thanks to the members of the mailing list desport²⁰. The authors also would like to thank Mihály Szilágyi and Louis Joseph Mattia for the proofreading of the paper.

References

- [AAB⁺16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg,

¹⁹<https://www.facebook.com/groups/udprog/>

²⁰<https://groups.google.com/forum/#!forum/desport-desamu>

- Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.
- [Aut17] The TensorFlow Authors. Computation using data flow graphs for scalable machine learning <http://tensorflow.org>. <https://github.com/tensorflow/tensorflow>, 2017. Online; accessed 22 January 2017.
- [Bát11] Norbert Bátfai. Do the object oriented programs have a mother tongue: or an introduction of an analytical weaving (*Van-e az objektum-orientált programoknak anyanyelve? avagy egy analitikai szövéssé bevezetése*). *Híradástechnika*, LXVI(2):27–32, 2011. Original paper in Hungarian.
- [Bát17a] Norbert Bátfai. Robopsychology. <https://github.com/nbatfai/Robopsychology/files/169195/robopsychology.pdf>, 2017. Online; accessed 2 February 2017.
- [Bát17b] Norbert Bátfai. Samu Entropy: ESPORT AND ARTIFICIAL INTELLIGENCE. <https://github.com/nbatfai/SamuEntropy>, 2017. Online; accessed 6 January 2017.
- [BB17] Norbert Bátfai and Renátó Besenczi. Robopsychology manifesto: Samu in his prenatal development. *submitted manuscript*, 2017.
- [BBL⁺17] Norbert Bátfai, Márió Bersenszki, Miklós Lukács, Renátó Besenczi, Gergő Bogacsovics, and Péter Jeszenszky. The common future of e-sport and robopsychology (*Az e-sport és a robotpszichológia közös jövője*). *submitted manuscript*, 2017. Original paper in Hungarian.
- [BBS16] N. Bátfai, R. Besenczi, and V. Simkó. *Samu Entropy developer’s guide*. <https://github.com/nbatfai/SamuEntropy/tree/master/docs>, 2016. translators: A. Fodor, G. Bogacsovics, J. Pogány.
- [BGNR16] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016.
- [CTB⁺10] Seth Cooper, Adrien Treuille, Janos Barbero, Andrew Leaver-Fay, Kathleen Tuite, Firas Khatib, Alex Cho Snyder, Michael Beenen, David Salesin, David Baker, and Zoran Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG ’10, pages 40–47. ACM, 2010.
- [CV05] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

- [Gol16] Peter Goldsborough. A tour of TensorFlow. *CoRR*, abs/1610.01178, 2016.
- [Kah74] G. Kahn. The semantics of a simple language for parallel programming. In *Information processing*, pages 471–475. North Holland, Amsterdam, 1974.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [KSSMB16] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4873–4882, 2016.
- [LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LCL⁺04] Ming Li, Xin Chen, Xin Li, Bin Ma, and P.M.B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.
- [LLWT15] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015.
- [LMHR⁺16] Erik Learned-Miller, Gary B. Huang, Aruni RoyChowdhury, Haoxiang Li, and Gang Hua. Labeled faces in the wild: A survey. In Michal Kawulok, M. Emre Celebi, and Bogdan Smolka, editors, *Advances in Face Detection and Facial Image Analysis*, pages 189–248. Springer International Publishing, 2016.
- [Lov94] László Lovász. *Algoritmusok bonyolultsága*. Nemzeti Tankönyvkiadó, 1994.
- [McC76] McCabe, T. J. A complexity measure. *IEEE Trans. Softw. Eng.*, 2(4):308–320, 1976.
- [MKS⁺15] V Mnih, K Kavukcuoglu, D Silver, AA Rusu, J Veness, MG Bellemare, A Graves, M Riedmiller, AK Fidjeland, G Ostrovski, S Petersen, C Beattie, A Sadik, I Antonoglou, H King, D Kumaran, D Wierstra, S Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.
- [NW14] H. W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *IEEE International Conference on Image Processing (ICIP)*, pages 343–347, 2014.
- [PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

- [Pen16] Roger Penrose. *The Emperor's new mind: Concerning computers, minds, and the laws of physics*. Oxford University Press, 2016.
- [PHS15] Marisa Ponti, Thomas Hillman, and Igor Stankovic. Science and gamification: The odd couple? In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play, CHI PLAY '15*, pages 679–684. ACM, 2015.
- [Sch44] Erwin Schrödinger. *What is life?: the physical aspect of the living cell*. Cambridge University Press, 1944.
- [SHM⁺16] D Silver, A Huang, CJ Maddison, A Guez, L Sifre, G van den Driessche, J Schrittwieser, I Antonoglou, V Panneershelvam, M Lanctot, S Dieleman, D Grewe, J Nham, N Kalchbrenner, I Sutskever, T Lillicrap, M Leach, K Kavukcuoglu, T Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
- [SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.
- [SLJ⁺14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [Sou12] T. B. Sousa. Dataflow programming concepts, languages and applications. http://paginas.fe.up.pt/prodei/ds12/papers/paper_17.pdf, 2012. Online; accessed 12 January 2017.
- [SVI⁺15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *CoRR*, abs/1512.00567, 2015.
- [Swa09] James Swallow. *SGU Stargate Universe*. Fandemonium Books, New York City, 2009.
- [TSBW16] Olivier Tremblay-Savard, Alexander Butyaev, and Jérôme Waldispühl. Collaborative solving in a human computing game using a market, skills and challenges. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play, CHI PLAY '16*, pages 130–141. ACM, 2016.
- [TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '14*, pages 1701–1708, 2014.
- [VSR⁺16] Arso M Vukicevic, Miroslav Stojadinovic, Milos Radovic, Milena Djordjevic, Bojana Andjelkovic Cirkovic, Tomislav Pejovic, Gordana Jovicic, and Nenad Filipovic. Automated development of

- artificial neural networks for clinical purposes: Application for predicting the outcome of choledocholithiasis surgery. *Computers in Biology and Medicine*, 75:80–89, 2016.
- [YLLL14] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.
- [ZL16] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.
- [ZZWS12] X. Zhang, L. Zhang, X. J. Wang, and H. Y. Shum. Finding celebrities in billions of web images. *IEEE Transactions on Multimedia*, 14(4):995–1007, 2012.